ISSN(Print): 2287-5905

http://dx.doi.org/10.3745/KTSDE.2012.1.2.075

A Study for Influence Measurement of Error by Quantitative Analysis

Eun-Ser Lee†

ABSTRACT

There are many problems that cause the process improvement of software and hardware, personality errors during software development. This paper propose the quantitative analysis of error that removes and manages the system problems as well.

Keywords : Error Recovery, Risk Management, Quantitative Analysis

정량적 분석에 의한 오류의 영향 측정에 관한 연구

이 은 서†

요 약

소프트웨어 개발 시, 프로세스 개선에 저해 요인이 되는 소프트웨어, 하드웨어, 인적 오류의 문제가 다수 존재한다. 시스템의 저해 요인을 제거하고 동시에 체계적으로 이를 관리하기 위하여 본 논문에서는 오류의 정량적인 분석방안을 제안한다.

키워드 : 오류회복, 위험관리, 정량적 분석

1. 서 론

소프트웨어 검토는 우리가 분석, 설계 및 코딩이라 불러온 소프트웨어 엔지니어링 활동을 정화시킨다. 검토는 소프트웨어 개발 동안 다양한 지점에 적용되며 오류들과 결함들을 발견하는 일을 한다[1].

다양한 유형의 검토가 소프트웨어 엔지니어링의 일부로 시행될 수 있고, 각각이 자신의 장소를 가지고 있다. 고객, 관리자, 기술직 직원들에게 소프트웨어 설계를 공식적으로 발표하는 것도 검토의 한 형태다[2]. 이외에 워크스루나 인스펙션도 정식 기술 검토로 인식되고 있다. 정식 기술 검토는 품질 보증 관점에서 볼 때 가장 효과적인 필터가 된다. 소프트웨어 엔지니어를 위해 소프트웨어 엔지니어에 의해 수행되는 정식 기술 검토는 소프트웨어 품질을 개선시키기 위한 효과적인 수단이다[9].

소프트웨어 프로세스 관점에 소프트웨어가 실사용자에게 배포된 후에 발견된 품질 문제를 의미하는 것이 결함이 된다. 오류도 소프트웨어가 실사용자에게 배포되기 전에 소프트웨어 엔지니어에 의해 발견된 품질 문제를 가리킨다. 품질에 영향을 미치지 않게 하기 위하여 각 사항들이 위험요소로 전이되는 것을 예방하고 예측하여 품질관리를 하려는 노력이 대두되고 있다. 따라서 본 논문에서는 발생된 오류를 식별하고 원인을 제거하는 기준 및 정량적인 방법을 제시하고자 한다.

2. 기본 개념

2.1 결함

품질은 많은 의미가 담긴 다차원 개면이다. 이로 인하여 파생되는 두 가지 중요한 결과가 있는데 하나는 소프트웨어 품질은 하나의 숫자로 축약할 수 없다는 것이다. 둘째는 품질 개념은 프로젝트마다 다르다. 아주 민감한 프로젝트에서 신뢰성은 가장 중요할 수 있지만 사용용이성은 그리 중요하지 않다. 소프트웨어 개발 프로젝트마다 개발하기 전에 품질 목표를 설정하여야 하고 개발 프로세스의 목표도 품질 목표를 만족시켜야 한다[5][10].

많은 품질 요인이 있지만 일반적으로 신뢰성이 소프트웨어 품질의 기준을 대표한다. 소프트웨어가 신뢰성이 없다는 것은 소프트웨어에 결함이 존재하기 때문이다. 품질을 측정하는 한 가지 방법은 출시된 소프트웨어의 단위 크기 당 결함의 수이다. 이러한 측면에서 볼 때 품질 목표는 할 수 있는 한 KLOC 당 결함의 수를 줄이는 것으로 귀결된다. 소프

※ 이 논문은 2012학년도 안동대학교 산학연구비에 의하여 연구되었음.
† 종신회원 : 안동대학교 컴퓨터공학과 부교수
논문접수 : 2012년 9월 24일
심사완료 : 2012년 10월 8일
* Corresponding Author : Eun-Ser Lee(eslee@andong.ac.kr)

트웨어 공학의 가장 좋은 기법으로 KLOC 당 1개 미만으로 줄일 수 있다[6].

품질에 대한 정의를 사용하려면 결함을 명확히 정의하여야 한다. 결함이란 소프트웨어가 고장 나게 하거나 출력이 바르지 않게 하는 문제를 말한다. 무엇을 결함으로 간주 할 것인지는 프로젝트나 프로젝트를 진행하는 기관이 사용하는 표준에 따라 다르다.

2.2 위험분석

위험이 소프트웨어 프로젝트에만 한정된 것은 아니다. 소프트웨어가 성공적으로 개발되고 고객에게 인도된 후에도 위험은 일어날 수 있다. 이러한 위험은 보통 현장에서의 소프트웨어 실패와 관련이 있다[7].

비록 잘 공학화된 시스템이 실패할 확률이 낮다고 해도, 컴퓨터-기저 제어 또는 감시 시스템에서 검출되지 않은 결점은 막대한 경제적 손실을 초래하거나 사람이 부상하거나 죽음을 초래할 만큼 더욱 나쁘고 심각할 수 있다. 그러나 컴퓨터-기저 제어와 감시 시스템의 비용상의 이점과 기능상의 이점은 보통 이러한 위험보다 증요하다. 오늘날 컴퓨터 소프트웨어와 하드웨어가 안전을 증시하는 시스템을 제어하기 위해 사용되고 있다.

소프트웨어 안정성과 위험분석은 소프트웨어에 부정적으로 영향을 미치는, 그리고 전체 시스템을 작동하지 않게 만드는 잠재적인 위험의 식별과 평가에 초점을 맞춘 소프트웨어 품질보증활동이다. 만약 소프트웨어 공학 프로세스의 초기에 치명적 위험을 식별할 수 있다면, 소프트웨어 설계 특징들을 명시할 수 있고, 잠재적인 치명적 위험을 제거하거나 조정할 수 있다.

2.3 품질 요소

소프트웨어의 품질을 결정하는 요소는 여러 가지가 있다. 소프트웨어의 결함, 원시코드에 포함된 오류는 소프트웨어의 품질을 결정하는데 매우 중요한 요소이다. 소프트웨어의 제작 과정 중 분석 및 설계와 구현 단계에서 유입되는 오류는 소프트웨어의 운용에 장애를 가져와 심각한 경제적 손해와 인명의 피해를 가져 올 수도 있다. 이러한 점에서 소프트웨어 제품에서 품질은 매우 중요한 요소이다[8].

소프트웨어에 대한 작업 관점이 어디 있느냐에 따라 품질 요소의 관심이 달라진다. 소프트웨어를 업그레이드하는 작업을 한다면 유지 보수성, 융통성, 테스트 용이성이 중요하다. 반면 플랫폼을 바꾸는 작업에는 이식성, 재사용성, 상호운용성이 중요한 요소일 것이다. 단지 소프트웨어를 운용한다면 정확성, 신뢰성, 효율성 등이 시스템의 품질을 결정한다[9].

소프트웨어의 품질의 특성은 세 가지 차원이 있다. 사용자에 의한 외부 관점을 나타내는 품질 요소 차원이 있다. 사용성, 신뢰성, 효율성 등 외부 품질이 여기에 속한다. 다음은 개발자 측면의 내부 관점을 나타내는 품질 기준이 있다. 마지막은 품질을 제어하기 위한 메트릭 차원이다. 메트릭은 품질 기준별로 측정 방법과 스케일 등을 정의하여 정확히 품질 기준을 측정할 수 있게 한다. 따라서 본 논문에서는 위와 같은 위험요소의 연관성을 정량적으로 분석하기 위하여 연구를 수행하고자 한다.

3. 본론 및 사례연구

본 장에서는 소프트웨어를 개발하는 과정에서 발생하는 오류를 찾아 분석하고 처리하는 방법을 제시하고 있다. 개발 과정으로는 요구사항 분석, 설계, 구현, 시험, 유지보수 단계가 있으며, 각 단계에서 발생할 수 있는 오류를 찾아서 이를 수정하기 위한 활동이 필요하다. 따라서 이와 같은 활동을 위하여 오류를 수집하고 분류 및 수정하여 실제 구현물에 개선된 사항을 적용할 수 있도록 해야 한다. 본 장에서는 오류 항목들의 수집, 오류 항목들의 분류, 오류 항목들의 수정, 오류 항목들의 개선으로 문제를 해결하고자 한다.

3.1 오류 항목들의 수집

소프트웨어를 개발하거나 프로젝트를 수행하는 과정에서는 무수히 많은 문제점들이 발생하게 된다. 이와 같은 문제점들은 해결이 되거나 결함으로 존재하여 향후 최종 산출물들이 구동되는 시기에 문제를 발생하게 된다. 따라서 발생하는 오류항목을 개발 과정 중에 모두 찾아내 처리를 하지 못한다면 잠재적인 오류에 의하여 문제가 발생하게 된다. 잠재적인 오류는 전체적인 시스템을 마비시키기도 하고 치명적인 오류를 유발시켜서 잘못된 결과를 산출할 수 있게 된다. 이와 같은 오류는 개발과정 뿐만이 아니라, 개발이 다 끝난 시점에서도 발생하여 동일한 문제를 발생시킨다.

발생된 오류 항목을 기초로 하여 발생 원인을 찾아서 다른 기능들에게 영향을 미치지 않도록 해야 한다. 따라서 오류 항목의 수집을 통하여 원인을 찾도록 노력해야 한다.

오류 항목의 수집을 위하여 수집 기준에 의하여 어떤 오류가 발생했는지 판단할 수 있어야 한다. 본 논문에서는 오류 항목의 기준을 오류의 발생단계별 수집, 오류의 진화단계에 따른 수집으로 분류하였다.

Table 1. Definition of error by progress phase of error

Phase	Contents
Initialization Phase Error	Error is occur to the development initialization phase (Development environment, tool, initialization off person resource, Development methodology etc.)
Transferal Phase Error	Error is occur to the system phase (Data transfer, Design, Analysis of business function)
Phase Error완료 단계 오류	Error is occur to the coding phase
Completion Phase Error	Error is occur to the test and maintenance phase
Commercialization Phase Error	Incorrect running error

오류의 발생단계별 수집은 개발의 생명주기에 의하여 발생오류를 수집하게 된다. 따라서 발생된 오류의 현재 상황을 확인할 수 있게 되고, 문제를 해결하기 위한 접근 단계를 파악하게 된다. 본 논문에서는 개발 생명 주기를 폭포수 모델로 기초하였다. 따라서 요구사항분석, 설계, 구현, 시험, 유지보수로 나누어서 오류 항목을 수집하도록 하였다.

다른 관점으로는 오류의 진화 단계에 따른 수집으로 다음과 같이 체계화 하였다.

초기화 단계 오류는 개발을 위하여 요구되는 기반 사항들로서 향후 개발을 위하여 기초가 되는 내용에서 발생하는 오류이다. 개발을 하는 과정에서 필요한 소프트웨어, 하드웨어 환경구축, 개발하려는 비즈니스의 요구정의, 개발인력 할당, 개발 방법론의 정의는 필수적이다. 이와 같은 사항은 초기에 올바른 선택이 이루어져야 하며, 초기화 단계 오류는 후행 단계에서 치명적인 오류를 발생 시킬 수 있게 된다.

전이 단계 오류는 개발 환경을 기반으로 하여 시스템을 분석하는 과정에서 발생하는 오류이다. 이와 같은 오류는 소프트웨어 기능의 불일치와 의도하지 않는 기능을 수행할 수 있게 된다. 또한 이후 단계에 오류를 전이하게 된다.

완료 단계 오류는 구현과정이거나 구현이 완료된 이후에 발생하는 오류이다. 완료 단계 오류는 단순히 소스코드의 수정 및 그 원인과 오류를 전달한 다음 단계의 결과물을 동시에 수정해야 한다.

상품화 단계 오류는 모든 개발이 다 끝난 후에 주 기능 또는 부 기능이 제대로 작동하지 않는 경우의 오류를 의미한다. 상품화 오류는 리콜이나 교환의 결과를 초래하므로 기업의 이미지와 신뢰성에 치명적인 영향을 미치게 된다. 상품화 단계 오류는 시험과 유지보수 단계에서도 발견을 못했거나 수정을 하지 못한 상태가 된다.

마지막으로 오류의 심각도에 따라서 상태를 분류하였다.

Table 2. Definition of error by degree of serious

Phase	Contents
Major	Uncertain cause or can't modify and improve as for myself
Normal	Clear cause or can modify and improve as for myself
Minor	Not transfer other phases and can modify and improve as for myself

심각도는 3단계로 구분하였으며, 오류 발생 시 수정 및 개선을 어떻게 전개해 나갈 수 있는지에 따라서 구분을 하였다.

3.2 오류 항목들의 코드화

3.1절에서 오류 항목을 수집하였다. 수집된 오류 항목은 조치를 위하여 해당 부분으로 전달할 필요가 있다. 이와 같은 작업은 오류 분류가 선행되어야 해당 오류를 전달하여 수정할 수 있게 된다. 오류를 전달하기 위해서는 3.1절의 자연어 형태가 아닌 시스템에서 자동화하여 제어가 가능하도록 할 필요가 있다. 따라서 오류형태에 따른 코드화가 요구된다.

오류 코드화하기 위해서 발생된 오류를 감지하여 어떤 형태의 오류인지를 판별해야 한다. 따라서 이를 위해서는 오류의 카테고리가 분야 별로 정의가 선행되어야 한다. 선행되어야 할 오류 정의 매칭시키기 위하여 과제에서 활용하고 있는 하드웨어, 소프트웨어에 대한 별도코드를 정의한다.

하드웨어는 각 장치마다 고유번호를 유지하고 있어서 장치 식별을 위한 별도의 작업이 불필요하다. 그러나 각 장치마다 발생하는 오류의 심각도에 따라서 복구가 가능한지 여부에 대해서 개발자에게 제공되어야 이를 기반으로 하여 개발자는 해당 장치의 오류를 수정할 수 있다. 하드웨어 자체의 오류에 의하여 교체가 요구되는 상황에서 펌웨어 소프트웨어적으로 해결을 하는 노력을 줄일 수 있게 된다. 하드웨어 오류에 대한 큰 범주는 다음과 같이 정의될 수 있다.

Table 3. Hardware code for the error items classification

Code	Contents
Replacement (H01)	Can't restore and need hardware replacement by hardware problems
Renew of Firmware software (H02)	Solved the problems by modify of firmware software
Reinstall of Firmware software (H03)	Solved the problems by modify of reinstall of software

소프트웨어의 경우도 오류 항목 분류를 위한 코드화가 수행되어야 한다. 여기서 소프트웨어는 사람이 작성하는 문서를 의미한다. 문서로는 프로젝트가 시작되어서 생성되어지는 문서이며, 요구사항 문서부터 소스코드 및 유지보수 문서까지 포함한다. 따라서 인적자원에 대한 오류를 별도로 구분하지 않은 이유는 문서를 담당자가 작성하기 때문에 소프트웨어 오류에 인적자원에 대한 오류도 포함되게 된다. 그러므로 소프트웨어 오류를 확인하여 인적자원에 대한 오류도 같이 파악하고자 한다. 각 개발 단계에서 소프트웨어 오류가 발생하기 때문에 오류가 나타나기 시작하는 시점은 확인을 할 수 있다. 문제점은 발생된 소프트웨어 오류가 어느 정도의 심각도를 가지고 있는지 시스템에서 확인하여 이를 수정 및 보완작업이 필요하게 된다. 따라서 발생되는 소프트웨어 자체의 심각도를 측정하는데 중점을 두고 다음과 같이 분류하였다.

소프트웨어 코드는 네 개로 구분을 하였다. 불분명의 경우, 소프트웨어 오류가 명확하기는 하지만 오류의 원인과 현상이 동일한 상황에서 같은 오류가 발생되지 않아서 원인을 명확히 찾기 힘든 경우이다. 이 경우에는 여러 상황의 시험을 거쳐서 동일한 상황에서 같은 오류가 발생하는 조건을 찾아서 수정을 할 수 있도록 해야 한다. 구조적 오류의 경우, 소프트웨어의 프로그래밍과 기능의 오류가 아닌, 인터

Table 4. Software code for the error items classification

Code	Contents
Uncertain (S01)	Error is not modify by unclear cause of error
Structural error (S02)	Error is happen to between other functions
Identify and modify cause of error (S03)	Error is modify by clear cause of error
Simple error (S04)	Error is remove only by modify of single software

페이스 오류와 같은 구조적인 오류에 해당된다. 이 경우에는 결국 설계의 오류가 원인이 되어서 발생된 것으로 설계 문서까지 수정을 필요로 하게 된다. 확인 및 원인 수정의 경우, 발생된 오류와 원인이 명확하여 수정을 할 수 있는 것을 의미한다. 이 경우에는 추적성에 의하여 기능이 전파된 이전 단계의 작업까지 수정을 필요로 하게 된다. 단순 오류의 경우, 프로그램 언어 문법 오류나 다른 소프트웨어에 영향을 주지 않고 자체 소프트웨어 내에서 발생된 것을 의미한다.

3.3 오류 항목의 심각도 측정

3.1과 3.2절에서는 오류 항목을 수집하고 코드화 하였다. 오류가 발생되는 순간이후에는 오류가 얼마나 심각한 내용인지에 대하여 관리가 필요하다. 이와 같은 관리는 다른 기능이나 단계에 오류를 얼마나 심각하게 전이하는지를 관리하고 심각한 정도에 따라서 즉각적인 대응을 할 수 있게 된다. 따라서 발생된 오류의 심각도를 측정하기 위하여 수학적인 접근 방법을 사용하였다. 오류에 대한 심각도 산출 수식은 다음과 같다.

$$ESD(p,t) = \prod_{i=1}^{5} D(p,t) + SDI(c,i)$$

ESD (Error Serious Degree) : 위험정도 산출
p (Phase) : 단계
t (Type) : 형태
SDI (Serious Degree of Item) : 오류 항목의 심각 정도
c (Cause) : 오류의 원인을 식별 가능 유무
i (Improvement) : 발생된 오류에 대하여 수정 및 개선의 가능 유무

D의 함수는 전반적으로 오류의 발생 단계와 형태를 식별하기 위한 것이다. 세부적인 사항으로 p의 경우, 3.1절에서 제시한 5단계를 나타낸다. 초기화, 전이, 완료, 안정화, 상품화 단계를 표시하여 오류가 발생된 단계를 확인할 수 있다. 식별을 위하여 초기화는 initialization (I), 전이 Transition(T), 완료는 Completion(C), 안정화는 Stabilizing(S), 상품화는 Merchandizing(M) 단계로 표현하였다. 그리고 t는 오류의 형태를 나타내고 있다. 오류의 형태로는 하드웨어(H), 소프트웨어(S), 인적자원(P)에 의한 원인을 식별할 수 있는지를 나타낸다. 또한 하드웨어, 소프트웨어, 인적자원에 대한 카테고리의 식별의 경우, 각 항목의 세밀한 분류가 선행 되어야 한다. 식별과 형태를 서로 상호 영향을 주기 때문에 상호 연관성을 정의해야 한다. 정의된 내용에서 결과 산출의 경우, 가장 높은 위험도를 가진 항목은 0.9, 보통은 0.5, 가장 낮은 위험 항목은 0.3으로 산출하였다. 정의된 표는 다음과 같다.

Table 5. Table of interrelation for the analysis of degree of serious error

Phase	Type	Result
I	H	0.3
	S	0.5
	P	0.9
T	H	0.3
	S	0.5
	P	0.9
C	H	0.3
	S	0.9
	P	0.5
S	H	0.9
	S	0.5
	P	0.3
M	H	0.5
	S	0.9
	P	0.3

Table 5의 결과 값은 오류의 심각도를 산출하는데 가중치로 활용이 된다. 가중치 산출은 경험적인 자료를 기반으로 각 단계에서 오류가 많이 나오는 것에 중점을 두고 값을 산출하였다.

SDI의 함수는 발생된 오류 항목이 얼마나 심각한 지를 나타내고 있다. 이를 평가하기 위하여 발생된 오류가 수정 및 개선이 가능 유무와 이를 위하여 오류의 원인을 명확히 식별해야 한다. 따라서 SDI에서는 c와 i에 의하여 두 경우를 판별하여 심각한 정도를 측정하고자 한다.

Table 6. Identification of Cause of error items

c	Result
Identify the error items and Identify clear cause of error	0.1
Unclear the error items and Identify clear cause of error	0.3
Identify the error items and Unclear cause of error	0.5
Uncertain the error items and Unclear cause of error	0.7

Table 6는 발생된 오류 항목에 대하여 원인이 식별 가능한지를 판별하기 위한 표이다.

Table 7. Modifiy and improvement for the occurrence of error

i	Result
Completely modify and improve against happen to error items	0.1
If it is incompletely modify and improve, correct running against happen to error items	0.3
If it is incompletely modify and improve, happen to problem against happen to error items	0.5
If it is incompletely modify and improve, incorrect running against happen to error items	0.7

Table 7은 발생된 오류 항목에 대하여 수정 및 개선이 가능하여 기능 수행에 문제가 없는지를 판별하기 위한 표이다. 3.3절에서 제시된 수치 자료를 기반으로 하여 오류에 대한 심각도를 산출하고자 한다. Table 5와 Table 6에 의하여 오류 항목의 심각 정도를 나타내는데 그 결과는 c + I 로 산출하게 된다.

4. 사례 연구

3장에서는 오류에 대한 항목을 식별하고 이를 수정 및 개선하여 발생된 오류가 얼마나 영향을 미치는 가에 대하여 심각도 분석을 위한 기반 내용을 제시하였다. 본 장에서는 3장의 내용을 기반으로 하여 사례 연구를 수행하였다. 사례 연구를 위해서는 적용한 시스템의 환경정의를 하고자 한다.

- Galaxy A(Model :SHW-M100S)

Items	Standard and contents
CPU	TI OMAP3440
GPU	Power VS SGX530
Memory	384RAM/1GM Flash
Display	3.7inch AM OLED(800X480)
Multitasking	Support
Auto answering function	Support
Voice meno recoding	Support

- 개발 시스템 개요

본 어플리케이션은 개인 시간표 혹은 매너모드 변경이 되어져야할 시간을 테이블 형태의 UI로부터 입력받는다. 입력받은 일정은 사용자에게 보기 편하도록 구성되어야 하며, 일정은 요일별로 다르게 입력할 수 있다. 입력받은 매너모드 변경 시간표의 시간이 되면 사용자가 모드를 변경하지 않아도 자동으로 매너모드 변경이 되며 일정표를 다른 사용자에게 문자로 전송할 수 있도록 한다.

개발한 응용 프로그램에서는 다음과 같은 사항의 오류 자료가 산출되었다. 산출된 내용은 3장을 기반으로 하여 산출하였으며, Table 7과 같다.

Table 8. Error computation data of application program

No	Phase(p)	Type(t)	Result	Cause(c)	Improv(i)	Result	Error Serious Degree (ESD)
1	I	P	0.9	0.1	0.1	0.2	1.1
2	T	S	0.5	0.3	0.1	0.4	0.9
3	C	S	0.9	0.3	0.3	0.6	1.5
4	C	H	0.3	0.5	0.3	0.8	1.1

Table 8에 의하여 오류의 위험 정도를 산출하였다. 산출 결과에 의하면 완료단계의 소프트웨어서 가장 큰 위험정도가 산출되었다. 따라서 산출된 항목을 수정 및 개선하기 위한 우선순위를 높이는 것이 전체적인 시스템의 위험관리를 통한 완성도를 높일 수 있게 된다.

5. 결론 및 향후 연구 방향

본 논문에서는 오류가 발생되는 시점에서 이를 해결하기 위하여 오류의 위험 정도를 정량화하기 위하여 산출 방법 및 기준을 제시하였다. 산출 방법에 따라서 오류의 위험도를 정량화하여 산출하여 영향도가 높은 오류를 선행적으로 해결하여 전체적인 시스템의 완성도와 안정성을 얻을 수 잇게 된다.

향후 연구로는 제시된 기준을 개발 과제의 영역별로 세분화하여 적용할 수 있도록 제시할 필요가 있다. 또한 이를 지원하기 위하여 자동화 도구를 개발하여 오류 관리를 효과적 및 효율적으로 할 수 있도록 개발할 계획에 있다.

참 고 문 헌

[1] Yoon chung, Successful Software development methodogy, Life power press, 1999. 8.

[2] Huh won sil, System analysis and design, Hanbit media, 2006. 5.

[3] Software process improvement forum, KASPA SPI-7, 2002.

[4] McCall, P.K. richards and G.F. walters, “Factors in software quality”, Vol.1, 2 and 3. Springfield VA, AD/A-049-014/015/055, 1997.

[5] Wohlin, Runeson, “Defect content estimations from review data”, Proceedings international conference on software engineering ICSE pp.400-409, 1998.

[6] Gaffney, John, “Some models for software defect analysis”, Lockheed martin, 1996.

[7] L.hatton, “Is modularization always good idea”, Information and software technology, Vol.38, pp.719-721, 1996.

[8] B. compton and C. withrow, “Prediction and control of ada software defects”, J. systems and software, Vol.12, pp. 199-207, 1990.

[9] Roger s. pressman “Software engineering” Mcgraw-hill international edition, 1997.

[10] Choi eun man, Software engineering, Jungik publishing co, 2011.

이 은 서

e-mail : eslee@andong.ac.kr

2001년～현 재 ISO/IEC 15504 국제 선임 심사원

2004년 중앙대학교 컴퓨터공학과(박사)

2004년～현 재 임베디드 산업협회 전문 위원

2004년～현 재 한국정보통신기술협회 위원

2012년 4월～현 재 안동대학교 컴퓨터공학과 부교수

관심분야 : CBD, Formal method, Quality model, SPI(Defect Analysis)